Amendments to the Specification

Please replace the paragraph on Page 1, lines 5 - 7 with the following marked-up replacement paragraph:

-- The present invention is related to commonly-assigned U. S. Patent _____ (serial number ~~09/_____, filed~~ 09/849,848, filed concurrently herewith), which is entitled "Recycling Events to Take Advantage of Capabilities of a Management System". –

Please replace the paragraph on Page 7, lines 11 - 12 with the following marked-up replacement paragraph:

-- Figure 5 illustrates a sample rule that may be used, according to the present invention, to trigger execution of newly-defined or different capabilities of an EMS using previously-defined events; and –

Please replace the paragraph on Page 17, lines 4 - 19 with the following marked-up replacement paragraph:

-- As will be apparent, adding an inventory registration process to an existing computing installation and updating the inventory repository requires running an inventory agent on those machines that are intended to be inventoried, which can be a costly process. Also, often software and hardware are installed and un-installed outside of the scheduled inventory runs. By contrast, the [[The]] event recycling process which is facilitated by annotated events, as disclosed herein, automates the registration process using previously received installation status events which are stored in an event repository and annotated such that they will cause registration of their

corresponding software product into the inventory repository. This process may be summarized as follows. The existing messages in the event repository may be inspected, searching for those whose message identifier indicates that the message represents a successful software installation. By determining whether each such message contains the inventory registration annotation property or properties (as exemplified by element 330 of Fig. 3B), the event can be recycled to be reexamined by the EMS to take appropriate action(s). If the message does not contain the inventory registration information, then the annotation may be added, and the registration process may then be invoked. (Note that tasks which are specified in annotations may be invoked immediately, if desired, or the invocation may be delayed as in a batch-mode approach. This is discussed further below, with reference to Fig. 7.) --

Please replace the paragraph that begins on Page 23, line 3 and carries over to Page 24, line 5 with the following marked-up replacement paragraph:

-- The logic in Blocks 660 - 690 enables the annotation process to be performed upon already-stored events. This logic may be used to annotate previously received events, as described above, including re-annotation of events which have previously been annotated. The annotation process beginning at Block [[600]] 660 may be invoked in several ways, such as upon expiration of a timer, upon reaching a predetermined counter value which counts the number of events added to the repository since the prior execution of this annotation process, upon receipt of an interrupt, and so forth. In Block 665, an optional test is performed to see if any new capabilities have been added to the EMS since the prior execution of the annotation process. (For example, the EMS may maintain information about its own capabilities, such as the version,

release, and modification level of installed features. This information may then be checked programmatically and/or a watch thread may signal when a change occurs.) If not, then the processing of the annotation process may be halted until a subsequent invocation, as shown in Fig. 6 by returning control to Block 660. When there are new capabilities, or when the test is not implemented, Block 670 indicates that an event is retrieved from the repository 655. Block 675 checks to ensure that a qualified event was retrieved. (As one example, this test may comprise determining whether the update flag as illustrated at 350 in Fig. 3B is set on or off.) When no more qualified events are found in the repository, then the test in Block 675 has a negative result and the processing of the annotation process may be halted until a subsequent invocation, as shown in Fig. 6 by returning control to Block 660. Otherwise, Block 680 checks to see if any new capability is applicable to this event. If so, the event is annotated (Block 685) and re-stored in the event repository (Block 690). Processing then returns to Block 670 to continue evaluation of existing events in event repository 655. When the test in Block 680 has a negative result, on the other hand, then this event does not need to be annotated, and the annotation process proceeds to evaluate another event by returning control to Block 670. --

Please replace the paragraph on Page 24, lines 6 - 15 with the following marked-up replacement paragraph:

-- One type of annotation that may be performed by Block 685 is a re-annotation of an already-annotated message. This has been previously discussed with reference to programmatically created information to invoke unregistration processing for events that were previously annotated to invoke a registration task. As another example, the new task which is

being dynamically reflected in the events may be to unregister all Product X, version 1.5 installations from the backup/restore processing system. (Perhaps a site-wide decision has been made that the backup and restore processes should be optimized by omitting backup/restore procedures for this software product.) In this case, the processing of Block 685 preferably comprises changing the value of the slot 330 (see ~~Fig. 3B)or property~~ Fig. 3B) or property 410 (see Fig. 4A) to the task which will perform this unregistration, and setting the update flag to FALSE to indicate that this task has not yet been performed. --

Please replace the paragraph on Page 25, lines 15 - 19 with the following marked-up replacement paragraph:

-- When the test in Block 715 has a negative result, then Block 720 checks to see if the currently-evaluated property specifies a task to be executed (such as the inventory registration task shown at 410 of Fig. 4A). If this test has a positive result, then at Block 730, a task server or task handler may be invoked to perform the applicable task. For example, the executable code named "INVENTORY[[.]]Command" is invoked at this point for the example event in Fig. 4A. --

Please replace the paragraph on Page 28, lines 4 - 8 with the following marked-up replacement paragraph:

-- While the preferred embodiments of the present invention have been described, additional variations and modifications in those embodiments may occur to those skilled in the art once they learn of the basic inventive concepts. Therefore, it is intended that the appended

claims shall be construed to include [[both]] the preferred ~~embodiment~~ embodiments and all such variations and modifications as fall within the spirit and scope of the invention. –